Assertions

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org

Assertions

Help to:

- Catch bugs as close to the source as possible
- Document and enforce *assumptions* and *invariants*

Assertions

```
#include <assert.h>
```

```
assert(true/false expression);
```

If `boolean expression` is false:

- Program exits immediately
- Exit level is non-zero, like when `main` returns non-zero
- Message is printed like:

```
Assertion failed: (denom != 0), function main, file foo.c, line 95.
```

Assertions

Document your assumptions:

```c
int sum = a*a + b*b;
assert(sum >= 0);
```

```c
if(isalpha(c)) {
    assert(c >= 'A');
    printf("%d\n", c - 'A');
}
```

Assertions

`assert` is not for typical error checking

```
FILE* input = fopen("numbers.txt", "r");
if(input == NULL) {
    printf("Error: could not open input file\n");
    return 1; // indicate error
}
```

Assertions

If checking for bad user input, or something else that is *strange but not impossible*:

- Use `if`,
- print a meaningful message, and
- if you must exit, return non-zero from `main` to indicate failure

For conditions that imply *your program is incorrect*, use `assert`

Assertions

```c
#include <stdio.h>
#include <assert.h>

int main() {
    int n = 0;
    scanf("%d", &n);
    if(n == 0) {
        printf("n must not be 0\n");
        return 1;
    }
    int n_sq = n * n;
    assert(n_sq >= n); // if false, something's wrong
    float n_inv = 1.0 / n;
    printf("squared=%d, inverse=%0.2f\n", n_sq, n_inv);
    return 0;
}
```

Assertions

```
$ gcc assert_eg.c -std=c99 -pedantic -Wall -Wextra
$ echo 4 | ./a.out
squared=16, inverse=0.25

$ echo -2 | ./a.out
squared=4, inverse=-0.50

$ echo 0 | ./a.out
n must not be 0

$ echo 200000000 | ./a.out
Assertion failed: (n_sq >= n), function main,
    file assert_eg.c, line 12.
```

Due to overflow of `int`!

- We'll talk more about overflow later